A Proposal of Throughput-Based MPTCP Scheduler for WebQoE Improvement

Takeshi Kato and Yoshihiro Ito
Graduate School of Engineering, Nagoya Institute of Technology, Nagoya, Japan
Email: kato@en.nitech.ac.jp, yoshi@nitech.ac.jp

***Abstract*—This study proposes a new packet scheduler scheme for Multi-Path TCP to improve the user experience (WebQoE) of Web services. The proposed scheme suppresses fluctuations in MPTCP ' s quality of service (QoS) while at the same time improving QoS. This scheme estimates QoS fluctuations from the throughput measurement, and subflows that can suppress fluctuations are selected based on this estimation. The authors have implemented the proposed scheme and evaluated QoS by experiment. The experimental results show that the proposed scheme can suppress QoS fluctuations more than the existing MPTCP packet scheduler schemes in various environments, confirming the effectiveness of the proposed scheme.**

***Index Terms*—MPTCP, Packet Scheduler, QoS, QoE**

I. Introduction

A new protocol called MPTCP[1] has been standardized to utilize diversified networks efficiently as an alternative to TCP. MPTCP treats multiple TCP connections as subflow. Since MPTCP has the same functions as TCP, MPTCP's congestion control[2] and packet scheduler[3] significantly affect the QoS provided by MPTCP. Therefore we must consider appropriate congestion control and packet scheduler for MPTCP

Concerning the QoS of MPTCP, [4] shows that even if a high QoS is achieved, Web service's Quality of Experience(QoE) can decrease if the QoS fluctuation is significant. Webqoe will increase if the QoS fluctuation is slight, even if the QoS is low. Thus, the authors proposed a new congestion control scheme that suppresses throughput fluctuation to improve WebQoE in MPTCP and showed its effectiveness by experiment[4]. It should be stressed that, as mentioned above, the combination of congestion control and packet scheduler is essential for QoE in MPTCP, and even if a congestion control can suppress throughput fluctuation, some packet schedulers may degrade the effectiveness. However, a packet scheduler that can suppress throughput fluctuations has yet to be studied. Moreover,since there is a trade-off relationship between the suppression of throughput fluctuation and throughput[4], it is necessary to consider the suppression of QoS degradation and the suppression of QoS fluctuation.

This paper proposes a new MPTCP packet scheduler that can suppress QoS fluctuations based on the throughput and confirm its effectiveness through experiments in an actual environment. This paper is organized as follows. Section II gives an overview of MPTCP. Sections III and IV show proposed method and our experiments, respectively. Section V indicates the results. Finally we conclude the paper in Sect. VI.

II. MPTCP

MPTCP is being standardized as one of the next-generation transport layer protocols to solve the various problems of TCP. MPTCP utilizes multiple TCP connections as subflow. This allows MPTCP to increase the available bandwidth compared to TCP, improve availability, and achieve higher service quality.

MPTCP uses the same three-way handshake to establish a connection as TCP does. However, MPTCP's three-way handshake is accompanied by the MP_CAPABLE option to check whether the peer supports MPTCP. The communication is performed using standard TCP if the other party does not support MPTCP.

Like TCP, MPTCP has a congestion control function using a congestion window, which is a variable window, but MPTCP's congestion control differs from TCP's in that it has a congestion window for each sub-flow.

Currently, representative congestion control schemes in MPTCP include LIA (Linked Increases Algorithm)[5] and OLIA (Opportunistic Linked Increases Algorithm)[6] for the loss-based method and WVEGAS(Weighted VEGAS)[7] for the delay-based one. Reference [4] also proposes a new congestion control based on WVEGAS that suppresses throughput fluctuations and confirms its effectiveness by experiment.

When multiple subflows are available, MPTCP must select which subflow to send data to. This choice is made by the packet scheduler, which divides the data received by MPTCP from the application into segments of size MSS, the maximum segment length, and distributes them to each subflow.

III. Proposal

Our proposed schedule is described below. First, the throughput of each subflow during communication is estimated, and the mean of the estimated throughput after the start of the communication is obtained. Next, the estimated throughput is compared with the mean measurement value. Finally, the subflow with the throughput value closest to the mean is selected to suppress fluctuations. Note that the following equation is used to estimate the throughput.

Here, the following equation is used to estimate throughput.

$$Throughput = \frac{cwnd \times MSS}{RTT} \tag{1}$$

In Eq.(1), $Throughput$ is the estimated throughput, $cwnd$ is the congestion window size, MSS is the maximum segment

size, and RTT is the time between sending a segment and receiving an ACK.

In addition, as an action to suppress the decrease in throughput, when the value closest to the mean value is lower than the mean value, it is controlled whether to select this subflow or the subflow with a value higher than the mean value, i.e., whether to select the subflow that suppresses throughput variation or the subflow that improves throughput. In other words, it controls choosing a subflow that suppresses throughput fluctuations or a subflow that enhances throughput. When these two subflows are observed during communication, the subflow is selected according to a probability value that a user can determine. The probability of choosing a path that suppresses throughput fluctuation is $alpha$. By arbitrarily selecting $alpha$, users can decide whether they want to prioritize suppressing throughput fluctuation or improving throughput, depending on their service. The pseudo code of the proposed method is shown in Algorithm 1.

Algorithm 1 Pseudocode of Proposed Packet Scheduler

$Selected\_subflow \Leftarrow NULL$
$mindif\_\mu \Leftarrow \infty$

for each subflow k **do**
  $Throughput \Leftarrow \frac{cwnd \times MSS}{RTT}$
  if $Throughput > Ave\_Throuhgput$ **then**
    $dif \Leftarrow Throughput - Ave\_Throughput$
    if $dif < mindif$ **then**
      $mindif \Leftarrow dif$
      $Selected\_subflow \Leftarrow k$
      $High\_subflow \Leftarrow k$
    end if
  end if
  if $Throughput < Ave\_Throuhgput$ **then**
    $dif \Leftarrow Ave\_Throughput - Throughput$
    if $dif < mindif$ **then**
      $mindif \Leftarrow dif$
      $Selected\_subflow \Leftarrow k$
      $Low\_subflow \Leftarrow k$
    end if
  end if
end for

if $Low\_subflow \neq NULL$**and**$High\_subflow \neq NULL$ **then**
  if $Low\_subflowisselectedwith\alpha\%probability$ **then**
    $Selected\_subflow \Leftarrow Low\_subflow$
  else
    $Selected\_subflow \Leftarrow High\_subflow$
  end if
end if

IV. Experiments

Figure 1 shows the experimental environment. In Fig.1, subjects access a Web server from a Web client through a network emulator. We treat a map service widely used worldwide, such as Google Maps. The network emulator acts as a router with dummynet[9], a kernel module for controlling network traffic in FreeBSD, and changes the communication quality of each path by adding delay and packet losses to packets that pass through it. As a result, we can create various environments where the communication quality of each path is uniform or heterogeneous, with the communication quality of paths differing from each other. Since one of the purposes of this experiment is to confirm whether the proposed method can suppress throughput fluctuations in any environment, each path is congested by generating TCP traffic between the load client and the load server using Autobench[10].

To compare our proposed method and the other, the packet schedulers used in this experiment are the proposed one and the default MPTCP one. The two schedulers are implemented on Linux, and the experimenter changes the scheduler used on the web server side. This experiment uses 25, 50, and 75% probability of selecting the path. Congestion control is based on LIA, the MPTCP standard.

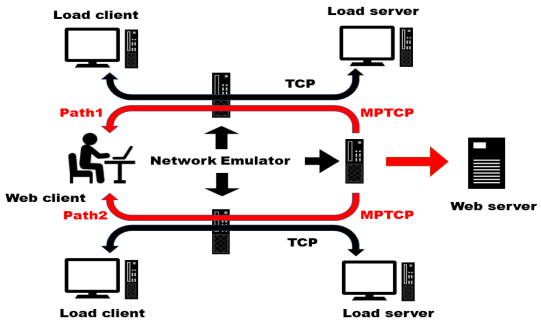

Fig. 1: Experimental Environment

TABLE I: Path specifications

Env.1 Env.2 Env.3			
	Bandwidth	Packet loss rate	delay
Path1	10Mb/s	3%	50ms
Path2	10Mb/s	3%	50ms
Env.4 Env.5 Env.6			
	Bandwidth	Packet loss rate	delay
Path1	10Mbps	1%	100ms
Path2	10Mbps	3%	50ms

TABLE II: Number of TCP connections

Env.1	Env.2	Env.3	Env.4	Env.5	Env.6
10	20	10∼20	10	20	10∼20

V. Results

Figures 2 through 5 show the experimental results. In these figures, the abscissa represents the experimental environment, the ordinate in Figs. 2 and 3 indicates the variance of throughput, and the vertical axes in Figs. 4 and 5 show the mean throughput. The 95% confidence intervals are also shown.

Figures 2 and 3 show that the throughput variance is lower when using our proposed scheduler compared to the default one in all six environments. This means that our proposal can suppress QoS fluctuations more than the default MPTCP

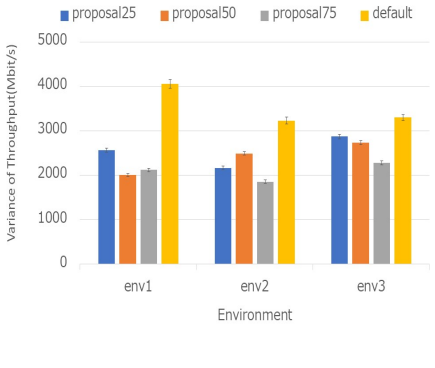

Fig. 2: Variance of Throughput (Env.1 through Env.3)

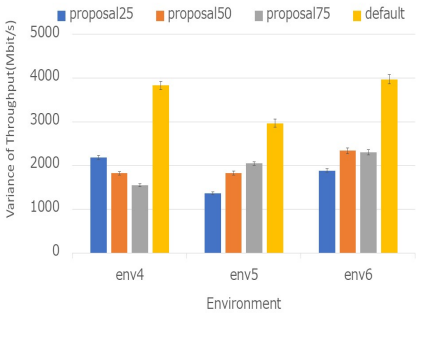

Fig. 3: Variance of Throughput (Env.4 through Env.6)

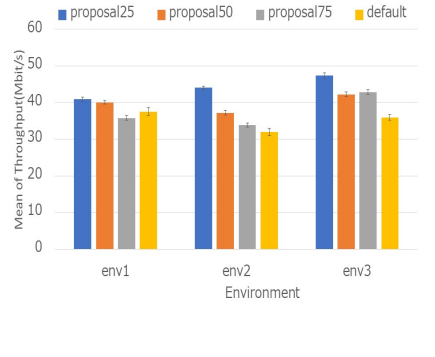

Fig. 4: Mean of Throughput (Env.1 through Env.3)

schedulers, even if the communication quality of each path is uniform or the communication quality differs.

From Figs. 4 and 5, we see that in all six environments, the mean of the throughput of our proposal is similar to that of the default scheduler. The proposed method not only suppresses throughput fluctuations better than the default scheduler but also suppresses throughput degradation. This will lead to an improvement in WebQoE.

On the other hand, the difference in the QoS of the proposal concerning the difference in $alpha$ was not what we expected. We had initially hoped that the proposed scheme would perform best when $alpha$ was 75, but the performance varied depending on the experimental environment. There are several possible causes for this. For example, when selecting a path that improves throughput, it is easier to find a path that improves throughput when there are more paths than when there are only two paths. Therefore, the more paths there are for communication, the more significant the difference in the results when the value of $alpha$ is changed. Since the experiment was conducted using two paths, it is thought that the differences were not so apparent. In addition, although we used a map search service as the Web service in this experiment, other Web services may cause a difference in performance. We plan to verify these results in the future.

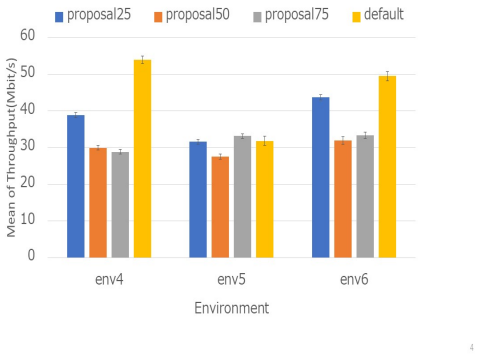

Fig. 5: Mean of Throughput (Env.4 through Env.6)

VI. Conclusions

In this study, we proposed a new packet scheduler that considers throughput as QoS and suppresses its fluctuation and throughput degradation. We also assessed the QoS through experiments, and the experimental results show the effectiveness of our proposal.

References

[1] A. Ford et al, "TCP Extensions for Multipath Operation with Multiple Addresses," IETF RFC 6824, Jan 2013.
[2] M. Welzl, "Network Congestion Control: Managing Internet Traffic,"2005.
[3] "Multi-Path TCP – Linux Kernel Implementation,"http://multipath-tcp.org
[4] T. Kato and Y. Ito, "A Proposal of Throughput-Based Congestion Control of MPTCP for WebQoE Improvement," Proc. 2022 IEEE 11th Global Conference on Consumer Electronics (GCCE), 2022.
[5] C. Raiciu and M. Handley and D. Wischik"Coupled Congestion Control for Multipath Transport Protocols,"RFC 6356 (Experimental),Oct 2011.
[6] Khalili, Ramin and Gast, Nicolas and Popovic, Miroslav and Upadhyay, Utkarsh and Le Boudec, Jean-Yves,"MPTCP is Not Pareto-optimal: Performance Issues and a Possible Solution,"Proc. 8th International Conference on Emerging Networking Experiments and Technologies,2012.
[7] Yu Cao and Mingwei Xu and Xiaoming Fu,"Delay-based Congestion Control for Multipath TCP,"Network Protocols (ICNP), 2012 20th IEEE International Conference on,Oct 2012.
[8] "Google(search service)"https://www.google.co.jp/maps
[9] L. Rizzo, "Dummynet," http://info.iet.unipi.it/luigi/dummynet/.
[10] "Autobench," http://www.xenoclast.org/autobench/.